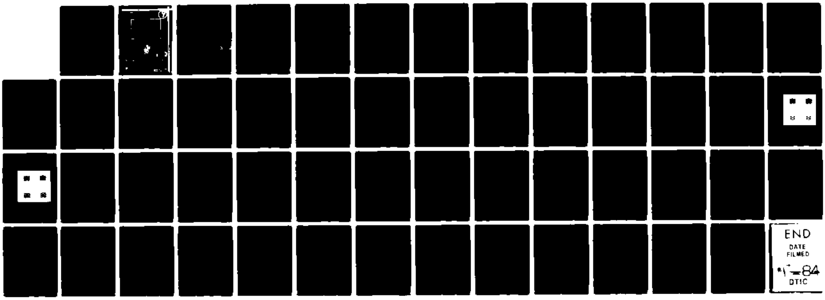
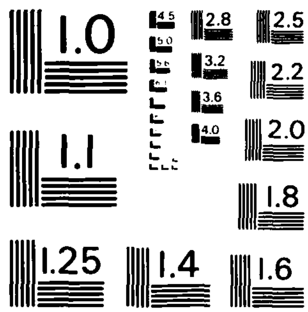
AD-A136 215 RECURSIVE INTERPOLATION OF SPACE-LIMITED SCENES(U) 1/1
VIRGINIA POLYTECHNIC INST AND STATE UNIV BLACKSBURG
DEPT OF ELECTRICAL ENGINEERING A A BEEX JUL 83
UNCLASSIFIED AFOSR-TR-83-1125 AFOSR-82-0234 F/G 12/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

AFOSR-TR- 83-1125

Virginia Polytechnic Institute and State University

Electrical Engineering
BLACKSBURG, VIRGINIA 24061

FINAL REPORT TO AFOSR

RECURSIVE INTERPOLATION OF SPACE-LIMITED SCENES

A. A. (Louis) Beex

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-1125	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) RECURSIVE INTERPOLATION OF SPACE-LIMITED SCENES		5. TYPE OF REPORT & PERIOD COVERED FINAL, 16 JUN 82-15 JUN 83
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) A.A. (Louis) Beex		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0234
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electrical Engineering Department Virginia Polytechnic Institute & State University Blacksburg VA 24061		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB DC 20332		12. REPORT DATE JUL 83
		13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Recursive reconstructions; space-limited scenes; noisy projections; robustness; convergence; soft constraints.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problem of extrapolation of scene-limited functions was investigated with an eye towards accomodating noisy measurements, and incorporating all available a priori information. An iterative projection approach was introduced, aimed specifically at removing a priori information and constraint incompatibility due to noisy measurements. Soft frequency domain measurement constraints and soft scene domain limitation constraints were proposed, and can alleviate convergence problems that occur when the imposed information is not compatible. The soft frequency domain measurement constraint specifically allows (CONTINUED)

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: the algorithm iterate to deviate from the noisy measurements, in recognition of the fact that the measurements were noisy, and they should therefore not be imposed as an absolute or hard constraint. The resulting alternating projection algorithm was shown to correspond to a non-expansive operator so that many solutions exist, all of which satisfy the a priori information and constraints.

FINAL REPORT

RECURSIVE INTERPOLATION OF SPACE-LIMITED SCENES

Prepared by: Dr. A. A. (Louis) Beex

Assistant Professor

Department of Electrical Engineering

Virginia Polytechnic Institute & State University

Blacksburg, VA 24061

Sponsored by: Air Force Office of Scientific Research

Air Force Systems Command

Directorate of Mathematical and Information Sciences

Bolling AFB, DC 20332

Report Number: 1

Contract Number: AFOSR - 82 - 0234

Research Period: 6/16/82 - 6/15/83

Program Manager: William R. Price, MAJ, USAF

Date: July 1983.

CONTENTS

RESEARCH OBJECTIVE

The research performed, and reported on herein, is a continuation of research performed at Rome Air Development Center, Griffiss AFB, during the Summer of 1981 under the USAF-SCEEE Summer Faculty Research Program |1|. This research is therefore directed at the enhancement of resolution in scenes with limited support. The importance to the Air Force lies in the application to space based infrared sensors. In this context we have a field of view limited by a sunshade, operated on by a Fourier transforming lens, and subsequently sampled over a finite support by a detector array.

The idea is to reconstruct the limited field of view scene, most compatible with the given information. This information consists of frequency domain samples, ultimately arrived at by measurements and therefore corrupted by noise, in addition to a priori information. The a priori information yields constraints on solutions to the problem, by such requirements as the nonnegativity of the scene function, the known physical extent of the scene function, and probabilistic characterizations of the corrupting noise.

The objective of the proposed research then, is to evaluate the performance of iterative deconvolution

algorithms, in the context of, and with the constraints for, the space based infrared sensor application. The sensitivity of the solutions with respect to the various constraints will indicate the robustness of the algorithms against assumptions made.

Until now iterative reconstruction algorithms did not incorporate effective ways of dealing with measurement noise. Consequently, the noisy measurements obtained in a practical application, could be incompatible with the a priori constraints. This then leads to searching for a nonexisting solution to the formulated problem. In this stage of the research we concentrate, therefore, on developing modifications of the original algorithm that explicitly allow the use of knowledge about the noise corrupting the measurements. Analysis of the resulting "soft constraint" algorithm shows that the essential convergence properties are preserved. Initial simulation results show the robustness of the new algorithm, when compared to the traditional "hard constraint" version.

RESEARCH STATUS

1. INTRODUCTION

In our problem an optical sensor has a sunshade that limits the field of view. The limited scene is subsequently Fourier transformed and a number of samples is taken in the Fourier domain. These frequency domain measurements are naturally subject to measurement noise. The idea now is to recover as much as possible of the original scene.

Philosophically at least this is a reasonable idea due to the analogy with the problem of extrapolating bandlimited signals. Several iterative projection algorithms have been proposed [2], and proofs of convergence have been given for the continuous problem [3,4]. The successful extrapolation by a factor of 20, in [5], led the author to extend and apply that computationally efficient one-shot approach to the 2-D problem outlined above [1]. The results of that approach were not encouraging due to an extreme sensitivity to noise, as well as the difficulty of implementing all a priori information. An iterative projection algorithm [6] facilitates the incorporation of a priori information at the expense of an increased computational burden. We develop a modification of some of the constraints, to accomodate the effects of measurement noise.

2. PROBLEM FORMULATION

As extensions to 2-D are rather elementary, 1-D formulations are used for transparency of the derivations.

Let $f(x)$ denote the continuous space-limited scene, and $F_c(\omega)$ its Fourier transform. The sequence $\{F_n\}$ is obtained from $F_c(\omega)$ by sampling with interval Ω. A periodic function $f_s(t)$, period 2π, can be associated with $\{F_n\}$ as follows

$$f_s(t) = \sum_{n=-\infty}^{\infty} F_n e^{+jtn} \tag{1}$$

For the Fourier coefficients we have, therefore,

$$F_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f_s(t) e^{-jtn} dt \tag{2}$$

As $\{F_n\}$ was obtained by sampling, the following relationship is valid

$$\begin{aligned} F_n &= F_c(n\Omega) \\ &= \int_{-\infty}^{\infty} f(x)\, e^{-jn\Omega x} dx \\ &= \sum_{r=-\infty}^{\infty} \int_{(2r-1)\pi/\Omega}^{(2r+1)\pi/\Omega} f(x) e^{-jn\Omega x} dx \end{aligned}$$

Substituting $\bar{x} + \frac{2\pi r}{\Omega}$ for x yields

$$F_n = \sum_{r=-\infty}^{\infty} \int_{-\pi/\Omega}^{\pi/\Omega} f(\bar{x}+\frac{2\pi r}{\Omega}) e^{-jn\Omega\bar{x}} e^{-jn\Omega 2\pi r/\Omega} d\bar{x}$$

Recognizing the last exponential to have a power which is an integer multiple of 2π, and substituting t for $\bar{x}\Omega$, results in:

$$F_n = \int_{-\pi}^{\pi} \sum_{r=-\infty}^{\infty} f((t+2\pi r)/\Omega)\; e^{-jnt}\; dt/\Omega \tag{3}$$

Comparing (2) and (3) yields the relationship between the scene-limited function f(.) and the periodic function $f_s(.)$.

$$f_s(t) = \frac{2\pi}{\Omega} \sum_{r=-\infty}^{\infty} f((t+2\pi r)/\Omega) \qquad (4)$$

The corresponding 2-D result follows.

$$f_s(t_1,t_2) = \frac{4\pi^2}{\Omega_1\Omega_2} \sum_{r_1=-\infty}^{\infty} \sum_{r_2=-\infty}^{\infty} f((t_1+2\pi r_1)/\Omega_1,\ (t_2+2\pi r_2)/\Omega_2)$$

Assume that f(x) is scene-limited to $[x_1,x_2]$, an interval of length x_2-x_1. From (4) the original function f(x) can be recovered from the periodic function $f_s(t)$ associated with the frequency domain samples, if those samples are spaced closely enough, that is

$$\Omega < \frac{2\pi}{x_2-x_1} \qquad (5)$$

The important result is, that the space-limited scene can be recovered from all frequency domain samples, under the constraint of (5). In the sequel (5) will be assumed valid by some margin, so called oversampling, so that $f_s(t)$ is truly scene-limited to a subset of $[-\pi,\pi]$.

It is a real world problem that all frequency domain samples are necessary. For the space based infrared sensor this requirement translates into an infinitely large detector array, which is clearly out of the question. In practice, therefore, we hope to get access to the dominant portion

of $\{F_n\}$, by means of an extrapolation process, so that

$$f_s(\tau) = \sum_{n=-\infty}^{\infty} F_n \tau^{-n} \tag{6}$$

$$\approx \sum_N F_n \tau^{-n} \tag{7}$$

A uniform sampling of $f_s(\tau)$ on the unit circle in the complex τ-plane, yields an N-sequence, $\{f_n\}$ say, for which

$$f_k = f_s(\tau)/_{\tau = W_N^{-k}}$$

$$= \sum_{n=-\infty}^{\infty} F_n W_N^{kn} \tag{8}$$

where

$$W_N \triangleq e^{-j2\pi/N} \tag{9}$$

Now a 1-to-1 relation exists between the N-sequence $\{f_n\}$ and the N-sequence $\{\tilde{F}_N\}$.

$$F_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k W_N^{-kn} \tag{10}$$

A substitution of (8) into (10) yields

$$\tilde{F}_n = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{m=-\infty}^{\infty} F_m W_N^{km} W_N^{-kn}$$

$$= \sum_{m=-\infty}^{\infty} F_m \left[\frac{1}{N} \sum_{k=0}^{N-1} W_N^{-k(n-m)} \right]$$

$$= \sum_{m=-\infty}^{\infty} F_m \, \delta_{m-n-rN} \qquad \forall \text{ integer } r$$

$$= \sum_{r=-\infty}^{\infty} F_{n+rN} \tag{11}$$

Consequently (7) will be an increasingly better approximation as N becomes large enough for the r=0 term in (11) to dominate. The accuracy of the detail in the reconstructed scene depends on N and the degree to which (7) is satisfied.

A graphical representation of the problem is shown in Figure 1.

3. SOLUTION APPROACH

A. Alternating Projection Algorithm

For the derivation and analysis that follows (similar to [6]) we assume the discrete frequency sequence $\{F_n\}_{-\infty}^{\infty}$ and the associated periodic function $f_s(t)$ over $[-\pi,\pi]$ to constitute our problem. Only a small number of the frequency domain samples is available due to a frequency domain distortion or truncation operator $\mathcal{T}$.

$$Y = \mathcal{T} F \tag{12}$$

The aim now, is to use all available a priori information to extrapolate, or rather estimate, the frequency domain samples. If such estimation is successful, a better estimate of the original is achieved than would be possible from the measurements without extrapolating estimation.

In addition to (12) a solution F must satisfy a number of constraints in the scene domain, indicated by a compound operator C_s. Also a constraint in the frequency domain exists, and is denoted by C_F. Consequently, we have

$$F = \mathcal{F} C_s \mathcal{F}^{-1} C_F F \tag{13}$$

or equivalently

$$F = C_F \mathcal{F} C_s \mathcal{F}^{-1} F \tag{14}$$

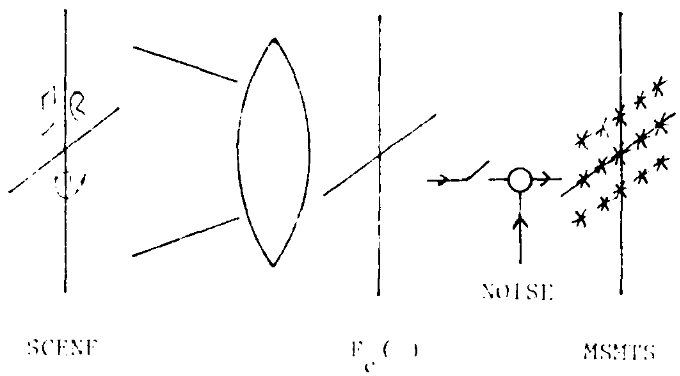


Figure 1. Problem Representation.

For notational convenience, let's define a constraint operator C encompassing (13) and (14), so that

$$F = CF \tag{15}$$

From (12) and (15) the following equality is valid, for any μ

$$F = CF + \mu\,(Y - TCF)$$

$$= \mathcal{O}F \tag{16}$$

$$= \mu Y + (I - \mu T)CF \tag{17}$$

The problem is to find a solution F to the above, or in other words, to find a so-called fixed point of the operator $\mathcal{O}$.

A common mode of attack is the successive approximation approach, implemented by

$$F_{k+1} = \mu Y + (I - \mu T)CF_k \tag{18}$$

According to the contraction mapping theorem, (18) yields a unique solution if $\mathcal{O}$ is a contraction operator, i.e., if $0 \leq \alpha \leq 1$ exists, such that

$$||\mathcal{O}F_1 - \mathcal{O}F_2|| \leq \alpha ||F_1 - F_2|| \;\forall\, F_1, F_2 \tag{19}$$

If α can take on the unit value, (19) defines a nonexpansive operator. A nonexpansive operator therefore decreases the distance between signals. The equivalent requirement for convergence of (18), is that $(I-\mu T)C$ is a contraction opera-

tor. For the latter it is sufficient that $I-\mu A$ and C are both nonexpansive, and one of the two is a contraction operator. As the second term on the righthand side of (18) constitutes a correction term over the measurement domain, one often encounters the choice $\mu=1$, expressing no need for correcting μY over that domain. We'll see shortly that in the present approach such a choice is not applicable.

B. Soft Constraints

One of the most crucial issues in having a chance at establishing some kind of convergence for the iterative process in (18), is the compatibility of the set of constraints that applies to the solution. As physical measurements are invariably subject to noise effects, it seems ill-advised to impose these noisy measurements as a hard constraint, i.e. as if these constituted absolute knowledge. As a matter of fact, such a hard constraint may lead to an incompatible constraint set, as demonstrated by the following example from the spectral estimation arena.

Suppose a small number of low-lag values of a covariance sequence is available, and extrapolation of these is desired in order to increase spectral resolution. A theoretical constraint on the spectral density function would be nonnegativity. Now suppose that due to measurement noise the zero lag covariance value is not the largest in magnitude. This now violates one of the necessary properties for covariance sequences, and results in an incompatibility of the given covariance sequence segment with any spectral density function.

In order to accomodate errors in the available measurements, we introduce the following soft constraint as frequency domain measurement constraint.

$$C_F F = \begin{cases} F & \text{if } ||F-Y||_M = \delta \leq \varepsilon \\ \left.\begin{array}{ll} Y + \frac{\varepsilon}{\delta}\,(F-Y) & \text{over } M \\ F & \text{over } M^c \end{array}\right\} & \text{if } ||F-Y||_M = \delta > \varepsilon \end{cases} \tag{20}$$

Herein, ε^2 indicates the tolerable, and δ^2 indicates the actual, mean square difference between reconstruction and measurements. These evaluations are made over the measurement domain M only. If corrective action takes place, the following relationship holds

$$||C_F F-Y||_M = ||Y + \frac{\varepsilon}{\delta}\,(F-Y)-Y||_M$$

$$= \frac{\varepsilon}{\delta}\,||F-Y||_M = \varepsilon$$

Therefore, we find,

$$||C_F F-Y||^2 = ||F-Y||^2_{M^c} + ||C_F F-Y||^2_M$$

$$\leq ||F-Y||^2_{M^c} + \frac{\varepsilon^2}{\delta^2}\,||F-Y||^2_M$$

$$\leq ||F-Y||^2 \tag{21}$$

which says that the iterate either remains unchanged, or moves closer to the measurements. Our soft measurement constraint leaves well enough alone, i.e. if the iterative process comes close enough to Y, with respect to the expect-

ed noise level, then the iterate is considered acceptable. An alternative soft constraint would be one that places a tolerated maximum on deviations from the measurements, for each measurement sample individually. Note that in any case a weighted norm is easily accomodated, so that known amplitude-and/or frequency-dependent noise information may readily be incorporated.

Several constraints apply in the scene domain. As the solution to our problem is a picture, compatible with our information, a nonnegativity operator applies, which is not softened. The scene is also of limited extent and it is argued that softening may be beneficial here. A border region B is proposed (similarities with [7]), to serve as a transition region between the known extent K, and the zeroed region Z. The soft scene limitation operator S then becomes

$$SF^{-1}F = \begin{cases} F^{-1}F & \text{over K} \\ g_B F^{-1}F & \text{over B} \\ 0 & \text{over Z} \end{cases} \tag{22}$$

where $0 \leq g_B < 1$ is a function defined over the border region, which provides a smooth transition to zero.

In the border region itself one could elect not to make any substitutions, and then monitor the energy in the border region. This information can potentially be used to accelerate the convergence of the algorithm. We hope to investigate this issue in the subsequent stage of research.

C. Convergence Issues

In order to assess the prospects for convergence of (19) the contraction properties of several operators must be evaluated [6,8]. For the scene limitation operator S, via Parseval's relationship

$$||SF^{-1}F_i - SF^{-1}F_j||^2 = \frac{1}{2\pi}\int_K |f_i - f_j|^2 dt + \frac{1}{2\pi}\int_B |g_B|^2 \cdot |f_i - f_j|^2 dt$$

$$= ||F^{-1}F_i - F^{-1}F_j||^2 \cdot r^2 \qquad (23)$$

where

$$r^2 = 1 - \frac{\frac{1}{2\pi}\int_Z |f_i - f_j|^2 dt + \frac{1}{2\pi}\int_B (1-|g_B|^2)|f_i - f_j|^2 dt}{||F^{-1}F_i - F^{-1}F_j||^2} \qquad (24)$$

The expression for r^2 is nonnegative and smaller or equal one, where equality occurs if f_i and f_j are equal over the union of B and Z. The operator S is therefore nonexpansive.

The nonnegativity operator can similarly be shown to be nonexpansive.

$$||PF^{-1}F_i - PF^{-1}F_j||^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |Pf_i - Pf_j|^2 dt$$

$$\leq \frac{1}{2\pi}\int_{-\pi}^{\pi} |f_i - f_j|^2 dt$$

$$\leq ||F^{-1}F_i - F^{-1}F_j||^2 \qquad (25)$$

Equality holds if the functions sign (f_i) and sign (f_j) are equal almost everywhere $[-\pi, \pi]$. The composite scene domain operator

$$C_S = PS$$

$$= SP \tag{26}$$

is, consequently, a nonexpansive operator.

The soft frequency domain measurement constraint yields,

$$||C_F F_i - C_F F_j||^2 = ||C_F F_i - C_F F_j||_M^2 + ||C_F F_i - C_F F_j||_{M^c}^2 \tag{27}$$

$$\leq ||F_i - F_j||_M^2 + ||F_i - F_j||_{M^c}^2 \tag{28}$$

$$\leq ||F_i - F_j||^2 \tag{29}$$

The inequality arises because the constraint leaves the components of F_i and F_j in M *alone, or moves one or both components* toward Y, and consequently closer together. The components in M^c are always unchanged. We find the measurement operator C_F, therefore, to be a nonexpansive operator also.

Rests us to evaluate the operator $(I-\mu T)$.

$$||F_i - \mu T F_i - F_j + \mu T F_j||^2 = ||F_i - \mu F_i - F_j + \mu F_j||_M^2 + ||F_i - F_j||_{M^c}^2 \tag{30}$$

$$= (1-\mu)^2 ||F_i - F_j||_M^2 + ||F_i - F_j||_{M^c}^2 \tag{31}$$

$$\leq ||F_i - F_j|| \quad \text{for } 0 < \mu < 2 \tag{32}$$

Equality holds if F_i equals F_j over M.

The conditions for equality in (24), (25), (29), and (32) could all be met at once, and consequently we find $(I-\mu T)C$

to be a non-expansive operator. As a result (18) may have many fixed points. It should be recognized that in view of the noisy measurements, a unique solution cannot be expected. Any solution, however, must satisfy all the constraints. The size of the solution set can possibly be reduced by decreasing ϵ in (20), but his moves the problem towards incompatibility of the available information.

D. Algorithm Simplification

Using the soft frequency domain constraint in (20), and the composite scene domain constraint in (26), in the algorithm formulation of (18), yields the following algorithm.

$$F_o = \mu Y$$

$$F_{k+1} = \mu Y + (I - \mu T) C_F \mathcal{F} C_S \mathcal{F}^{-1} F_K \tag{33}$$

This algorithm is represented in Figure 2.

Let's define

$$\tilde{F}_k = C_F \mathcal{F} C_S \mathcal{F}^{-1} F_k \tag{34}$$

so that the right hand side of (33) can be written and simplified as follows.

$$\tilde{F}_k - \mu T \tilde{F}_k + \mu Y = \tilde{F}_k \quad \text{over } M^c \tag{35}$$

$$= (1-\mu) F_k + \mu Y \quad \text{over } M \tag{36}$$

The latter equality can be rewritten as

$$\tilde{F}_k - \mu T \tilde{F}_k + \mu Y = Y + (1-\mu)(\tilde{F}_k - Y) \tag{37}$$

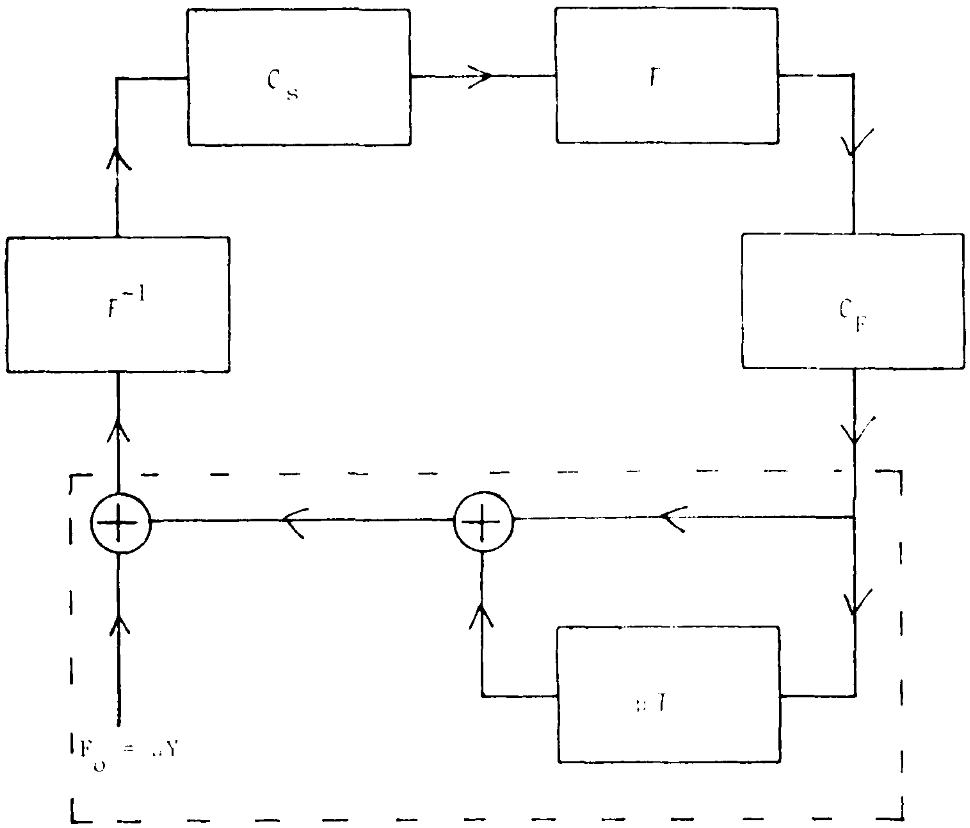


Figure 2. Algorithm Simplification.

If we now substitute $\frac{\varepsilon}{\delta}$ for $(1-\mu)$ then (36) and (37) can be seen to represent the soft frequency domain constraint of (20). As $\frac{\varepsilon}{\delta}$ is smaller or equal one the corresponding choice for μ:

$$\mu = 1 - \frac{\varepsilon}{\delta} \tag{38}$$

is seen to satisfy the convergence conditions of (31). As a result of making this particular choice for μ, we would have in Figure 2, two identical consecutive frequency domain constraint operators. The algorithm therefore simplifies because the dot-enclosed algorithm part now has the effect of an identity operator, due to the equality

$$C_F \cdot C_F = I \cdot C_F \tag{39}$$

The resulting algorithm simply consists of transformations from frequency domain to scene domain and vice versa, with an application of the respective constraint operators.

Note that if our tolerable noise level equals zero, i.e. $\varepsilon = 0$ in (20), then (20) as well as (38) cause the algorithm to degenerate to the traditional hard-constraint algorithm, in which the iterate is replaced with the measurements over the measurement domain.

4. SIMULATIONS

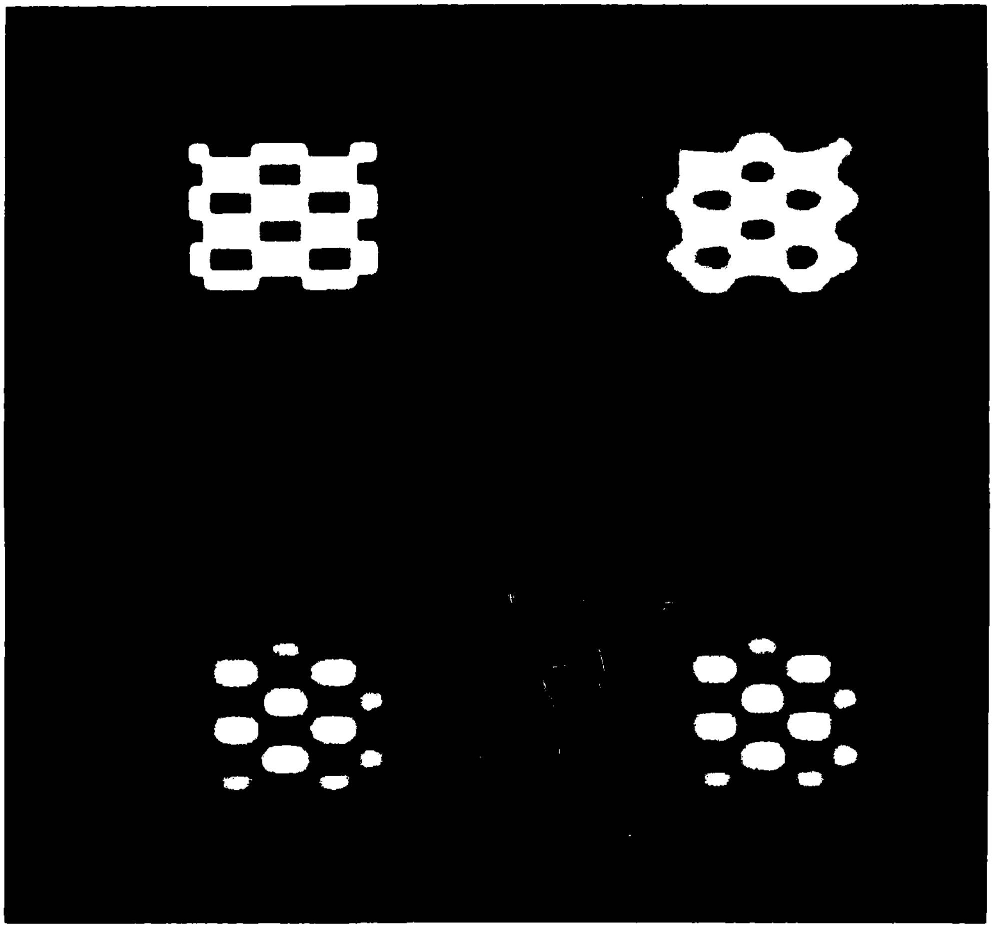
For each of the blocks in Figure 2 commands were written in RATFOR, for subsequent running on the VAX 11/78o in VPI&SU's SDA lab. A command is like a self-contained subroutine called from the keyboard, and usually operating on existing files. Each iteration of the algorithm therefore takes four commands. One more command was written to take care of all initializations. Codes for these commands can be found in the Appendix.

Our scene domain consists of a 64 x 64 pixel array. After generating a checkerboard pattern the scene domain was constrained to the center 25 x 20 (hor x vert) pixels, which therefore constitute the region of known extent K. The resulting scene limited picture was then transformed and the measurement domain M was chosen to be the 15 x 19 (hor x vert) area centered at zero frequency. The mean square value of the frequency domain signal turned out to be 27 x 10^6. Noise can be added to these frequency domain measurement values. We used a noise variance of 40 x 10^3. The scene domain constraint implemented in these simulations left a scene of 27 x 22 (hor x vert) pixels, so that the border region B is 1 pixel wide on each side.

At present the set of commands has to be entered sequentially in order to effect a number of iterations of the algorithm. This time consuming process can be alleviated in the future by rewriting the commands into a

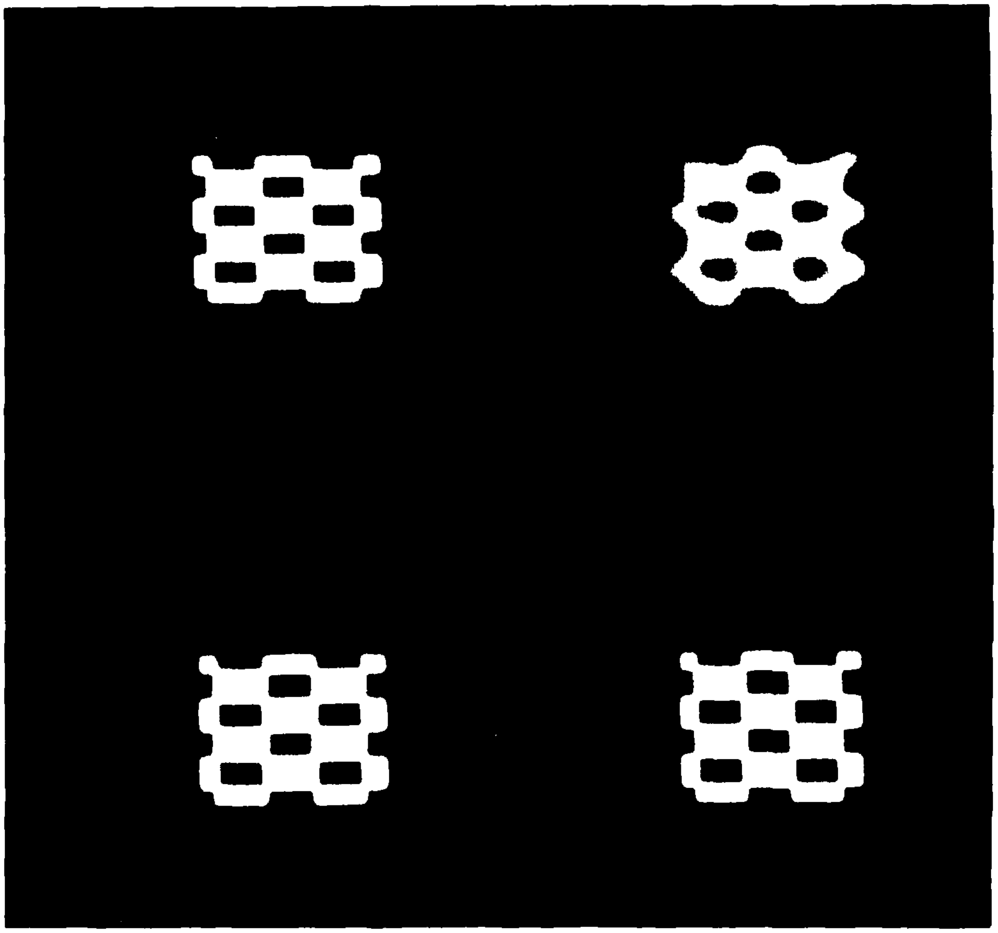
single command, which will then facilitate a more comprehensive study of the convergence behavior for large numbers of iterations of the present algorithm. Consequently our present numerical results are limited to 7 iterations of the algorithm. This limited experiment, however, demonstrates what is expected to be the characteristic behavior of the algorithm.

Starting from the initial condition without noise and going through 7 iterations will not lead to any dramatic results. We illustrate, nevertheless, the operational condition of the algorithm in Figure 3, where a substantial improvement in resolution is obtained after only seven iterations, when compared to a simple inverse transform of the measurements. To illustrate the characteristic behavior of the soft-constraint algorithm on the basis of only 7 iterations, the next experiment, with noisy measurements (SNR = 28 dB) starts from the other end. Let the true solution be the initial iterate, which should therefore constitute a feasible solution for the algorithm. The iterate F_k in the traditional algorithm with the hard frequency domain constraint then moves away from the true solution F, thereby indicating that this algorithm cannot possibly converge to the true solution. This effect was actually observable on the TV monitor, although it does not show in the photograph of Figure 4. We therefore derived numerical indicators of the behavior of the reconstructions,



Upper Left: Original Scene
Upper Right: Inverse of Measurements <u>without</u> Noise
Lower Left: Iterate 7 before Scene Domain Constraint Operator
Lower Right: Iterate 7 after Scene Domain Constraint Operator

Figure 3. Algorithm Operation for Noiseless Measurements, with Inverse of Measurements as Initial Interate.



Upper Left: Original Scene
Upper Right: Inverse of Measurements with Noise
Lower Left: Iterate 7 before Scene Domain Constraint Operator
Lower Right: Iterate 7 after Scene Domain Constraint Operator

Figure 4. Algorithm Operation for Noisy Measurements, with Original Scene as Initial Interate.

and these RMS values are represented in Figure 5. The iterate in the soft constraint algorithm with the noise variance underestimated at 10,000 (the noise variance equals 40,000) is seen to also move away from the true solution, but much slower. Finally, it is clear that if a good estimate of the noise variance is available, then the iterate will not move away from the true solution, because they are close enough already to satisfy the soft constraint. The specific effects of estimated and actual noise variance will be investigated in a subsequent research effort.

REFERENCES

1. A. A. Beex, "Enhanced Scene Resolution: 2-D Spectral Estimator Approaches," Final report 1981 USAF-SCEEE Summer Faculty Research Program, August 1981.

2. R. W. Gerchberg, "Super-resolution through Error Energy Reduction," Opt. Acta, Vol. 21, pp. 709-720, 1974.

3. A. Papoulis, "A new Algorithm in Spectral Analysis and Bandlimited Signal Extrapolation," CAS-22, pp. 735-742, 1975.

4. D. Youla, "Generalized Image Restoration by the Method of Alternating Orthogonal Projections," CAS-25, pp. 694-702, 1978.

5. J. A. Cadzow, "An Extrapolation Procedure for Bandlimited Signals," ASSP-27, pp. 4-12, 1979.

6. R. W. Schafer et al, "Constrained Iterative Restoration Algorithms," Proc. IEEE, Vol. 69, No. 4, April 1981.

7. T. F. Quatieri, D. E. Dudgeon, "Implementation of 2-D Digital Filters by Iterative Methods," ASSP-30, No. 3, June 1982.

8. V. T. Tom et al, "Convergence of Iterative Nonexpansive Signal Reconstruction Algorithms," ASSP-29, No. 5, October 1981.

/

```
#--FFTDRV                  DRIVER
#
#IDENTIFICATION
#           TITLE          FFTDRV
#           AUTHOR         AA(LOUIS) BEEX
#           VERSION        A.01
#           DATE           15 SEPT 1982
#           LANGUAGE       RATFOR
#           SYSTEM         VAX-11
#
#PURPOSE
#           DRIVER FOR COMMAND THAT PERFORMS THE RADIX-2 FAST
FOURIER #           TRANSFORM OF A REAL IMAGE
#
#ENTRY POINT
#           FFTDRV(WORK,ERRET)
#
#ARGUMENT LISTING
#           WORK        INT         WORK ARRAY
#           ERRET       INT         ALTERNATE ERROR RETURN
#
#INCLUDE FILES/COMMONS
#           MACA1       INCLUDE     GIPSY TOKEN DEFINITIONS FOR A1
CHARACTER #          GIPCOM      INCLUDE
#           ERRET       INCLUDE     INCLUDE FILE FOR COMMON ERROR
#
#ROUTINES CALLED
#           PPUSH       PUSHES PROGRAM NAME INTO ERROR STACK
(GIPSY) #           PPOP        POPS PROGRAM NAME FROM ERROR STACK
(GIPSY) #           RDKINL      OPEN A SIF AND FILL IN THE IDENT
BLOCK #          CLOSE       CLOSES A FILE(GIPSY)
#           FFTALG      COMPUTES THE (INVERSE) FAST FOURIER
TRANSFORM (USER) #
```

```
*******************************************************************
#
 INCLUDE MACA1
 SUBROUTINE FFTDRV(WORK,*)
 IMPLICIT INTEGER (A-Z)
 INCLUDE GIPCOM
 INCLUDE ERROR
 INTEGER IDENT(.IDLENGTH)
 DIMENSION WORK(.ARB)
#
```

```
 EQUIVALENCE(NPPL,IDENT(.IDNPPL))
 EQUIVALENCE(NLIN,IDENT(.IDNLINS))
 EQUIVALENCE(NCOL,IDENT(.IDNCOLS))
 EQUIVALENCE(NROW,IDENT(.IDNROWS))
 EQUIVALENCE(NBND,IDENT(.IDNBNDS))
 EQUIVALENCE(MODE,IDENT(.IDMODE))
#
 CALL PPUSH('FFTDRV')
#
#                                OPEN INPUT FILE
#
 CALL RDKINL(FDI1,IDENT,.OLD,IEV,%9999)
 CALL CLOSE(FDI1)
#
#                                CHECK INPUT FILE
#
 IF(MODE^=.REALMODE) GOTO 9000
 M=1
 NLIN2=NLIN/2
 WHILE(NLIN2>1)
 $(
 M=M+1
 NLIN2=NLIN2/2
 $)
 IF(NLIN^=2**M) GOTO 9020
 N=1
 NPPL2=NPPL/2
 WHILE(NPPL2>1)
 $(
 N=N+1
 NPPL2=NPPL2/2
 $)
 IF(NPPL^=2**N) GOTO 9020
 NMAX=MAX(NPPL,NLIN)
#
#
 NXT=1
 ILIN1=GETWP(NXT,.REALMODE,NPPL)
 ILIN2=GETWP(NXT,.REALMODE,NPPL)
 XAR=GETWP(NXT,.REALMODE,2*NMAX)
 FTAR=GETWP(NXT,.REALMODE,2*NLIN*NPPL)
#
 IF(.OK^=OSALOC(NXT)) GOTO 9010
#
 CALL COMTIN(%9999)
#
#                               CALL FFT ROUTINE
#
 CALL FFTALG(FDI1,FDO1,NBND,WORK(ILIN1),WORK(ILIN2),WORK(XAR),
WORK(FTAR),NLIN,NPPL,NMAX,IEV,%9999)
#
 CALL PPOP
#
 RETURN
#
#                               ERROR CONDITIONS
 9000 CONTINUE
#
```

```
#                                           ILLEGAL DATA MODE
 IEV=-2012
 GOTO 9999
#
 9010 CONTINUE
 IEV=OSGIEV(IEV)
 GOTO 9999
#
#                                           ILLEGAL ARRAY SIZE
 9020 CONTINUE
 IEV=-5004
 GOTO 9999
#
 9999 CONTINUE
 CALL CLOSE(FDI1)
 CALL CLOSE(FDO1)
#
 RETURN 1
 END
```

```
#--FFTALG                    I/O &ALGORITHM
#
#IDENTIFICATION
#       TITLE           FFTALG
#       AUTHOR          AA(LOUIS) BEEX
#       VERSION         A.01
#       DATE            15 SEPT 1982
#       LANGUAGE        RATFOR
#       SYSTEM          VAX-11
#
#PURPOSE
#       TO COMPUTE THE FAST FOURIER TRANSFORM (RADIX-2)
#       OF A SIF IMAGE (1 BND IF REAL, 2 BNDS IF COMPLEX)
#
#ENTRY POINT
#
FFTALG(FDI,FDO,NBND,ILIN1,ILIN2,XAR,FTAR,NLIN,NPPL,NMAX,IEV,*) #
#ARGUMENT LISTING
#       FDI          INPUT FILE DESCRIPTOR
#       FDO          OUTPUT FILE DESCRIPTOR
#       NBND          NO OF BANDS IN IMAGE TO BE PROCESSED
#       ILIN1        LINE BUFFER FOR REAL PART OF INPUT IMAGE
#       ILIN2        LINE BUFFER FOR IMAGINARY PART OF IMAGE
#       XAR          WORKARRAY
#       FTAR         WORKARRAY
#       NLIN         NO OF ROWS IN THE IMAGE
#       NPPL         NO OF COLUMNS IN THE IMAGE
#       NMAX         MAX OF NPPL AND NLIN
#       INV          INVERSE TRANSFORM IF TRUE
#       IEV          INTEGER EVENT VARIABLE
#       ERRET        ALTERNATE RETURN
#
#INCLUDE FILES/COMMONS
#       MACA1        INCLUDE (GIPSY)
#
#ROUTINES CALLED
#       PPUSH
#       PPOP
#       CLOSE
#       CPYIDR
#       COPYDS
#       RREAD
#       RWRITE
#       RX2FFT
#       DSCNAM
#


*******************************************************************
#
 INCLUDE MACA1
 SUBROUTINE
FFTALG(FDI1,FDO1,NBND,ILIN1,ILIN2,XAR,FTAR,NLIN,NPPL,NMAX,IEV,*
IMPLICIT INTEGER (A-Z)  CHARACTER FDI1(.FDLENGTH),
FDO1(.FDLENGTH)  INTEGER IDENT(.IDLENGTH), JDENT(.IDLENGTH)
 REAL ILIN1(NPPL), ILIN2(NPPL), NORM
 COMPLEX XAR(NMAX), FTAR(NLIN,NPPL)
 COMPLEX Z
 LOGICAL INV
#
 INCLUDE TTCOM
#
```

```
#
#
 CALL PPUSH('FFTALG')
#
 NMAX=MAX(NLIN,NPPL)
 FORWARD=1
 INVERSE=-1
 DFLT=1
 CALL RNGETI('FORWARD(1) OR INVERSE(-1)
TRANSFORM?.',INVERSE,FORWARD,DFLT, VAR,IEV,%9000)
 INV=.TRUE.
 IF(VAR==1) INV=.FALSE.
#                                  OPEN INPUTFILE
 CALL CPYIDR(FDI1,IDENT,.OPNTMP,IEV,%9000)
#
#                                  WRITE DESCRIPTOR RECORD TO TEMP
FILE  CALL DSCNAM('FFTALG',IEV,%9000)
#
#                                  SET UP OUTPUT FILE PARAMETERS
 JDENT(.IDNPPL)=IDENT(.IDNPPL)
 JDENT(.IDNLINS)=IDENT(.IDNLINS)
 JDENT(.IDNCOLS)=IDENT(.IDNCOLS)
 JDENT(.IDNROWS)=IDENT(.IDNROWS)
 JDENT(.IDNBNDS)=2
 JDENT(.IDMODE)=2
#
#AN INVERSE TRANSFORM MUST YIELD A REAL IMAGE FOR OUR
APPLICATION: #NO. OF OUTPUT BANDS EQUALS 1
 NOB=2
 IF(INV) NOB=1
 JDENT(.IDNBNDS)=NOB
#
#                                        OPEN OUTPUT FILE
 CALL COPYDS(FDO1,JDENT,IEV,%9000)
#
 DO BLKN=1,NLIN
 $(
 CALL RREAD(FDI1,ILIN1,1,BLKN,IDENT,.WAIT,IEV,%9000)
 IF(NBND==2)
   $(
   CALL RREAD(FDI1,ILIN2,2,BLKN,IDENT,.WAIT,IEV,%9000)
   $)
 ELSE
   $(
   DO I=1,NPPL
     $(
     ILIN2(I)=0.
     $)
   $)
 IF(INV)
   $(
   DO J=1,NPPL
     $(
     XAR(J)=CMPLX(ILIN1(J),ILIN2(J))
     XAR(J)=CONJG(XAR(J))
     $)
   $)
 ELSE
```

```
    $(
    DO J=1,NPPL
      $(
      XAR(J)=CMPLX(ILIN1(J),ILIN2(J))
      $)
    $)
  CALL RX2FFT(XAR,NPPL)
  DO J=1,NPPL
  $(
  FTAR(BLKN,J)=XAR(J)
  $)
  $)
  NORM=FLOAT(NLIN*NPPL)
  DO J=1,NPPL
  $(
    DO I=1,NLIN
      $(
      XAR(I)=(FTAR(I,J))
      $)
  CALL RX2FFT(XAR,NLIN)
  IF(INV)
  $(
  DO I=1,NLIN
    $(
    FTAR(I,J)=CONJG(XAR(I))/NORM
    $)
  $)
  ELSE
  $(
  DO I=1,NLIN
    $(
  FTAR(I,J)=XAR(I)
    $)
  $)
  $)
#
  DO BLKN=1,NLIN
  $(
  DO I=1,NPPL
    $(
    Z=FTAR(BLKN,I)
    ILIN1(I)=REAL(Z)
    ILIN2(I)=AIMAG(Z)
    $)
  CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
  IF(NOB==1) GOTO 1000
  CALL RWRITE(FDO1,ILIN2,2,BLKN,JDENT,.WAIT,IEV,%9000)
  1000 CONTINUE
  $)
  CALL PPOP
  CALL CLOSE(FDI1)
  CALL CLOSE(FDO1)
  RETURN
#
#
#                                            ERROR CONDITIONS
  9000 CONTINUE
  CALL CLOSE(FDI1)
```

```
 CALL CLOSE(FD01)
#
 9999 CONTINUE
 9090 RETURN 1
 END
```

```
#--RX2FFT      THIS IS A SUBROUTINE CALLED IN FFTALG.RAT
 SUBROUTINE RX2FFT(X,N)
 COMPLEX X(N), U, W, T
 NV2=N/2
 NM1=N-1
 M=1
 WHILE(NV2>1)
 $(
 M=M+1
 NV2=NV2/2
 $)
 NV2=N/2
 J=1
 DO I=1,NM1
 $(
 IF(I>=J) GOTO 5
 T=X(J)
 X(J)=X(I)
 X(I)=T
 5 K=NV2
 6 IF(K>=J) GOTO 7
 J=J-K
 K=K/2
 GOTO 6
 7 J=J+K
 $)
 PI=4.*ATAN(1.)
 DO L=1,M
 $(
 LE=2**L
 LE1=LE/2
 U=CMPLX(1.,0.)
 FLE1=FLOAT(LE1)
 CARG=COS(PI/FLE1)
 SARG=-1.*SIN(PI/FLE1)
 W=CMPLX(CARG,SARG)
 DO J=1,LE1
   $(
   DO I=J,N,LE
     $(
     IP=I+LE1
     T=X(IP)*U
     X(IP)=X(I)-T
     X(I)=X(I)+T
     $)
   U=U*W
   $)
 $)
 RETURN
 END
```

This subroutine is an adaptation of the Cooley, Lewis, and Welch algorithm for decimation-in-time, radix-2, in-place FFT.

```
#--SCNCTD                DRIVER
#
#IDENTIFICATION BEEX, RATFOR, VAX-11
#
#PURPOSE
#          DRIVER FOR COMMAND THAT ZEROES A SCENE
#          OUTSIDE OF A SPECIFIED WINDOW
#
#ENTRY POINT SCNCTD(WORK,ERRET)
#
#INCLUDE FILES/COMMONS
#          MACA1
#          GIPCOM
#          ERRET
#
#ROUTINES CALLED
#          PPUSH
#          PPOP
#          RDKINL
#          CLOSE
#          SCNCTA     ALGORITHM SECTION OF SCNCTT COMMAND
#
#*****************************************************************
#
 INCLUDE MACA1
 SUBROUTINE SCNCTD(WORK,*)
 IMPLICIT INTEGER (A-Z)
 INCLUDE GIPCOM
 INCLUDE ERROR
 INTEGER IDENT(.IDLENGTH)
 DIMENSION WORK(.ARB)
#
 EQUIVALENCE (NPPL,IDENT(.IDNPPL))
 EQUIVALENCE (NLIN,IDENT(.IDNLINS))
 EQUIVALENCE (NCOL,IDENT(.IDNCOLS))
 EQUIVALENCE (NROW,IDENT(.IDNROWS))
 EQUIVALENCE (NBND,IDENT(.IDNBNDS))
 EQUIVALENCE (MODE,IDENT(.IDMODE))
#
 CALL PPUSH('SCNCTD')
#
#                        OPEN INPUTFILE
#
 CALL RDKINL(FDI1,IDENT,.OLD,IEV,%9999)
 CALL CLOSE(FDI1)
#
#                      . CHECK INPUTFILE
#
 IF(MODE^=.REALMODE) GOTO 9000
#
 NP=NPPL*NLIN
 N=1
 NP2=NP/2
 WHILE(NP2>1)
 $(
   N=N+1
   NP2=NP2/2
 $)
 IF(NP^=2**N) GOTO 9020
 IF(NBND^=1) GOTO 9020
#
 NXT=1
```

```
 ILIN1=GETWP(NXT,.REALMODE,NPPL)
 IF(.OK^=OSALOC(NXT)) GOTO 9010
#
 CALL COMTIN(%9999)
#
#                          CALL ALGORITHM SECTION
#
 CALL SCNCTA(FDI1,FDO1,WORK(ILIN1),NPPL,NLIN,IEV,%9999)
#
 CALL PPOP
#
 RETURN
#                          ERROR CONDITIONS
 9000 CONTINUE
#                          ILLEGAL DATA MODE
 IEV=-2012
 GOTO 9999
#
 9010 CONTINUE
 IEV=OSGIEV(IEV)
 GOTO 9999
#                          ILLEGAL ARRAY SIZE
 9020 CONTINUE
 IEV=-5004
 GOTO 9999
#
 9999 CONTINUE
 CALL CLOSE(FDI1)
 CALL CLOSE(FDO1)
#
 RETURN1
 END
```

```
#--SCNCTA                          ALGORITHM
#
#IDENTIFICATION BEEX, RATFOR, VAX-11
#
#PURPOSE
#             ALGORITHM SECTION FOR COMMAND THAT ZEROES
#             A SCENE OUTSIDE OF A SPECIFIED WINDOW
#
#ENTRY POINT
#             SCNCTA(FDI,FDO,ILIN1,NPPL,NLIN,IEV,*)
#
#ARGUMENT LISTING
#             FDI     INPUTFILE DESCRIPTOR
#             FDO     OUTPUTFILE DESCRIPTOR
#             ILIN1   LINE BUFFER
#             NPPL    NO POINT PER LINE
#             NLIN    NO LINES
#             IEV
#
#***************************************************************
#
 INCLUDE MACA1
 SUBROUTINE SCNCTA(FDI1,FDO1,ILIN1,NPPL,NLIN,IEV,*)
 IMPLICIT INTEGER (A-Z)
 CHARACTER FDI1(.FDLENGTH), FDO1(.FDLENGTH)
 INTEGER IDENT(.IDLENGTH), JDENT(.IDLENGTH)
 REAL ILIN1(NPPL)
#
 INCLUDE TTCOM
#
 CALL PPUSH('SCNCTA')
#
 HLOW=2
 HHI=NPPL-1
 DFLT=NPPL/10
 CALL RNGETI('HORIZONTAL WINDOW
SIZE?.',HLOW,HHI,DFLT,WXLEN,IEV,%9000)   VLO=2
 VHI=NLIN-1
 DFLT=NLIN/10
 CALL RNGETI('VERTICAL WINDOW
SIZE?.',VLO,VHI,DFLT,WYLEN,IEV,%9000)   IF(WXLEN^=(WXLEN/2)*2)
WXLEN=WXLEN+1   IF(WYLEN^=(WYLEN/2)*2)  WYLEN=WYLEN+1
#
#                       OPEN INPUTFILE
#
 CALL CPYIDR(FDI1,IDENT,.OPNTMP,IEV,%9000)
 CALL DSCNAM('SCNCTA',IEV,%9000)
#
 DO I=6,19
 $(
 IF (I .EQ. 6 .OR. I .EQ. 7 .OR. I .EQ. 13 .OR. I .EQ. 14 .OR.
     I .EQ. 17 .OR. I .EQ. 19) $(
                                   X=IDENT(I)
                                   JDENT(I)=X
                                 $)
 $)
#                       OPEN OUTPUTFILE
 CALL COPYDS(FDO1,JDENT,IEV,%9000)
#
 IF(WXLEN<=0/WXLEN>=NPPL/WYLEN<=0/WYLEN>=NLIN) GOTO 9000
#
 XWM1=(NPPL-WXLEN)/2
```

```
 XWB=XWM1+1
 XWEP1=XWB+WXLEN
 XWE=XWEP1-1
#
 YWM1=(NLIN-WYLEN)/2
 YWB=YWM1+1
 YWEP1=YWB+WYLEN
 YWE=YWEP1-1
#
DO I=1,NPPL
 $(
   ILIN1(I)=0.
 $)
 DO BLKN=1,YWM1
 $(
   CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
 $)
 DO BLKN=YWEP1,NLIN
 $(
   CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
 $)
 DO BLKN=YWB,YWE
 $(
   CALL RREAD(FDI1,ILIN1,1,BLKN,IDENT,.WAIT,IEV,%9000)
   DO I=1,XWM1
   $(
     ILIN1(I)=0.
   $)
   DO I=XWEP1,NPPL
   $(
     ILIN1(I)=0.
   $)
#                           POSITIVITY CONSTRAINT
   DO I=XWB,XWE
   $(
     IF(ILIN1(I)<0.) ILIN1(I)=0.
   $)
   CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
 $)
#
 CALL PPOP
 CALL CLOSE(FDI1)
 CALL CLOSE(FDO1)
 RETURN
#
#                           ERROR CONDITIONS
#
 9000 CONTINUE
 CALL CLOSE(FDI1)
 CALL CLOSE(FDO1)
#
 9999 CONTINUE
 9090 RETURN 1
 END
```

```
# --FRQCTD                 DRIVER
#
#IDENTIFICATION BEEX, RATFOR, VAX-11
#
#PURPOSE
#     DRIVER SECTION FOR COMMAND THAT IMPOSES A
#     FREQUENCY DOMAIN MEASUREMENT CONSTRAINT
#
#ENTRY POINT FRQCTD(WORK,ERRET)
#
#INCLUDE FILES/COMMONS
#     MACA1
#     GIPCOM
#     ERRET
#
#ROUTINES CALLED
#     PPUSH
#     PPOP
#     RDKINL
#     CLOSE
#     FRQCTA  ALGORITHM SECTION OF FRQCTC COMMAND
#*******************************************************
#
 INCLUDE MACA1
 SUBROUTINE FRQCTD(WORK,*)
 IMPLICIT INTEGER(A - Z)
 INCLUDE GIPCOM
 INCLUDE ERROR
 INTEGER IDENT(.IDLENGTH), LDENT(.IDLENGTH)
 DIMENSION WORK(.ARB)
#
 EQUIVALENCE (NPPL,IDENT(.IDNPPL))
 EQUIVALENCE (NLIN,IDENT(.IDNLINS))
 EQUIVALENCE (NCOL,IDENT(.IDNCOLS))
 EQUIVALENCE (NROW,IDENT(.IDNROWS))
 EQUIVALENCE (NBND,IDENT(.IDNBNDS))
 EQUIVALENCE (MODE,IDENT(.IDMODE))
#
 CALL PPUSH('FRQCTD')
#                          OPEN INPUTFILES
 CALL RDKINL(FDI1,IDENT,.OLD,IEV,%9999)
 CALL CLOSE(FDI1)
 CALL RDKINL(FDI2,LDENT,.OLD,IEV,%9999)
 CALL CLOSE(FDI2)
#                          CHECK INPUTFILES
 IF(MODE^=.REALMODE) GOTO 9000
#NEED TO CHECK EQUALITY OF IDENT AND LDENT ARRAYS BUT NOT ALL
#
 NXT=1
 ILIN1=GETWP(NXT,.REALMODE,NPPL)
 ILIN2=GETWP(NXT,.REALMODE,NPPL)
 CORR=GETWP(NXT,.REALMODE,NLIN*NPPL)
 CORI=GETWP(NXT,.REALMODE,NLIN*NPPL)
#
 IF(.OK^=OSALOC(NXT)) GOTO 9010
#
 CALL COMTIN(%9999)
#
 CALL FRQCTA(FDI1,FDI2,FDO1,WORK(ILIN1),WORK(ILIN2),
            WORK(CORR),WORK(CORI),NLIN,NPPL,IEV,%9999)
#
 CALL PPOP
```

```
#
 RETURN
#                        ERROR CONDITIONS
 9000 CONTINUE
#                        ILLEGAL DATAMODE/INCOMPATIBLE INPUT ARRAYS
 IEV=-2012
 GOTO 9999
#
 9010 CONTINUE
 IEV=OSGIEV(IEV)
 GOTO 9999
#
 9999 CONTINUE
 CALL CLOSE(FDI1)
 CALL CLOSE(FDI2)
 CALL CLOSE(FDO1)
#
 RETURN 1
 END
```

```
#--FRQCTA            I/O & ALGORITHM
#
#IDENTIFICATION BEEX, RATFOR, VAX-11
#
#PURPOSE
#      ALGORITHM AND INPUT/OUTPUT SECTION FOR COMMAND THAT
#      IMPOSES A FREQUENCY DOMAIN MEASUREMENT CONSTRAINT
#
#ENTRY POINT
#      FRQCTA(FDI1,FDI2,FDO1,ILIN1,ILIN2,CORR,CORI,NLIN,NPPL,IEV,*)
#
#ARGUMENT LISTING
#      FDI1   MEASUREMENTS
#      FDI2   OLD RECONSTRUCTION
#      FDO1   NEW RECONSTRUCTION
#
#INCLUDE FILES/COMMONS
#      MACA1
#
#ROUTINES CALLED
#      PPUSH
#      PPOP
#      CLOSE
#      CPYIDR
#      COPYDS
#      RREAD
#      RWRITE
#      DSCNAM
#
#*********************************************************
#
 INCLUDE MACA1
 SUBROUTINE FRQCTA(FDI1,FDI2,FDO1,ILIN1,ILIN2,CORR,CORI,
                   NLIN,NPPL,IEV,*)
 IMPLICIT INTEGER (A - Z)
 CHARACTER FDI1(.FDLENGTH), FDI2(.FDLENGTH), FDO1(.FDLENGTH)
 INTEGER IDENT(.IDLENGTH), LDENT(.IDLENGTH), JDENT(.IDLENGTH)
 REAL ILIN1(NPPL), ILIN2(NPPL), SUM, THRESH, SOFCOR, DIF
 REAL CORR(NLIN,NPPL), CORI(NLIN,NPPL)
 REAL TL, TH, TD
#
 INCLUDE TTCOM
#
 CALL PPUSH('FRQCTA')
#
#                          OPEN INPUTFILES
 CALL CPYIDR(FDI1,IDENT,.OPNTMP,IEV,%9000)
 CALL CPYIDR(FDI2,LDENT,.OPNTMP,IEV,%9000)
#                          WRITE DESCRIPTOR RECORD TO TEMP FILE
 CALL DSCNAM('FRQCTA',IEV,%9000)
# CALL DSCNAM('FRQCTA',IEV,%9000)
#                          SET UP OUTPUTFILE PARAMETERS
 DO I=6,19
 $(
   IF (I .EQ. 6 .OR. I .EQ. 7 .OR. I .EQ. 13 .OR. I .EQ. 14 .OR.
       I .EQ. 17 .OR. I .EQ. 19) $(
                                   X=IDENT(I)
                                   JDENT(I)=X
                                 $)
 $)
#                          OPEN OUTPUTFILE
 CALL COPYDS(FDO1,JDENT,IEV,%9000)
```

```
#
 BLKN=NLIN/2+1
#                   READ THE INFO PASSED ALONG IN THE MSMT ARRAY
#                   LINES 1 THRU YS AND NLIN-YS+2 THRU NLIN FOR
#                   COLS 1 THRU XS AND NPPL-XS+2 THRU NPPL
#                   CONSTITUTE THE MSMT DOMAIN IN FREQUENCY SPACE
 CALL RREAD(FDI1,ILIN1,1,BLKN,IDENT,.WAIT,IEV,%9000)
 YS=INT(ILIN1(1))
 XS=INT(ILIN1(2))
#
#                   DETERMINE MEAN SQUARE DEVIATION OVER MSMT DOMAIN
#                   (EVENTUALLY ADD MAXIMUM DEVIATION OPTION ETC)
 DO I=1,NLIN
 $(
   DO J=1,NPPL
   $(
     CORR(I,J)=0.
     CORI(I,J)=0.
   $)
 $)
 SUM=0.
 DO BLKN=1,YS
 $(
   DO BND=1,2
   $(
     CALL RREAD(FDI1,ILIN1,BND,BLKN,IDENT,.WAIT,IEV,%9000)
     CALL RREAD(FDI2,ILIN2,BND,BLKN,LDENT,.WAIT,IEV,%9000)
     DO I=1,XS
     $(
       DIF=ILIN1(I)-ILIN2(I)
       SUM=SUM+DIF**2
       IF(BND==1) CORR(BLKN,I)=DIF
       IF(BND==2) CORI(BLKN,I)=DIF
     $)
     IF(XS>=1)
     $(
       NS=NPPL-XS+2
       DO I=NS,NPPL
       $(
         DIF=ILIN1(I)-ILIN2(I)
         SUM=SUM+DIF**2
         IF(BND==1) CORR(BLKN,I)=DIF
         IF(BND==2) CORI(BLKN,I)=DIF
       $)
     $)
   $)
 $)
 IF(YS>=1)
 $(
   MS=NLIN-YS+2
   DO BLKN=MS,NLIN
   $(
     DO BND=1,2
     $(
       CALL RREAD(FDI1,ILIN1,BND,BLKN,IDENT,.WAIT,IEV,%9000)
       CALL RREAD(FDI2,ILIN2,BND,BLKN,LDENT,.WAIT,IEV,%9000)
       DO I=1,XS
       $(
```

```
            DIF=ILIN1(I)-ILIN2(I)
            SUM=SUM+DIF**2
            IF(BND==1) CORR(BLKN,I)=DIF
            IF(BND==2) CORI(BLKN,I)=DIF
          $)
          IF(XS>=1)
          $(
            DO I=NS,NPPL
            $(
              DIF=ILIN1(I)-ILIN2(I)
              SUM=SUM+DIF**2
              IF(BND==1) CORR(BLKN,I)=DIF
              IF(BND==2) CORI(BLKN,I)=DIF
            $)
          $)
        $)
      $)
    $)
#                  MEAN SQUARE DIFFERENCE OVER MSMT DOMAIN
    SUM=SUM/((2*XS-1)*(2*YS-1))
#                  COMPARE WITH TOLERABLE MEAN SQUARE DIFFERENCE
    TL=0.0
    TH=1.E+08
    TD=1.
    CALL RNGETR('THRESHOLD MSV?.',TL,TH,TD,THRESH,IEV,%9000)
    IF(SUM<=THRESH) GOTO 5000
#                  SOFT CORRECTION
    SOFCOR=SQRT(THRESH/SUM)
    SOFCOR=1.-SOFCOR
    WRITE(TTYOT,1005) SOFCOR
    1005 FORMAT(' SOFCOR= ',E9.4)
#
    WRITE(TTYOT,1010) THRESH, SUM
    1010 FORMAT(' VARIANCE THRESHOLD= ',E9.4,' MS DIFF= ',E9.4)
    DO BLKN=1,YS
    $(
      CALL RREAD(FDI2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
      CALL RREAD(FDI2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
      DO I=1,XS
      $(
        ILIN1(I)=ILIN1(I)+SOFCOR*CORR(BLKN,I)
        ILIN2(I)=ILIN2(I)+SOFCOR*CORI(BLKN,I)
      $)
      IF(XS>=1)
      $(
        DO I=NS,NPPL
        $(
          ILIN1(I)=ILIN1(I)+SOFCOR*CORR(BLKN,I)
          ILIN2(I)=ILIN2(I)+SOFCOR*CORI(BLKN,I)
        $)
      $)
      CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
      CALL RWRITE(FDO1,ILIN2,2,BLKN,JDENT,.WAIT,IEV,%9000)
    $)
    IF(YS>=1)
    $(
      DO BLKN=MS,NLIN
      $(
```

```
      CALL RREAD(FDI2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
      CALL RREAD(FDI2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
      DO I=1,XS
      $(
        ILIN1(I)=ILIN1(I)+SOFCOR*CORR(BLKN,I)
        ILIN2(I)=ILIN2(I)+SOFCOR*CORI(BLKN,I)
      $)
      IF(XS>=1)
      $(
        DO I=NS,NPPL
        $(
          ILIN1(I)=ILIN1(I)+SOFCOR*CORR(BLKN,I)
          ILIN2(I)=ILIN2(I)+SOFCOR*CORI(BLKN,I)
        $)
      $)
      CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
      CALL RWRITE(FDO1,ILIN2,2,BLKN,JDENT,.WAIT,IEV,%9000)
    $)
  $)
#                   COPY OUTSIDE MEASUREMENT DOMAIN
    MSM1=MS-1
    YSP1=YS+1
    IF(MSM1>=YSP1)
    $(
      DO BLKN=YSP1,MSM1
      $(
        CALL RREAD(FDI2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
        CALL RREAD(FDI2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
        CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
        CALL RWRITE(FDO1,ILIN2,2,BLKN,JDENT,.WAIT,IEV,%9000)
      $)
    $)
  GOTO 6000
#                    NO CORRECTION, STRAIGHT COPY
  5000 CONTINUE
  WRITE(TTYOT,1020) THRESH
  1020 FORMAT(' THRESHOLD= ',E8.2,' SATISFIED')
  DO BLKN=1,NLIN
  $(
    CALL RREAD(FDI2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
    CALL RREAD(FDI2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
    CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
    CALL RWRITE(FDO1,ILIN2,2,BLKN,JDENT,.WAIT,IEV,%9000)
  $)
#
  6000 CONTINUE
#
  CALL PPOP
  CALL CLOSE(FDI1)
  CALL CLOSE(FDI2)
  CALL CLOSE(FDO1)
  RETURN
#                      ERROR CONDITIONS
  9000 CONTINUE
  CALL CLOSE(FDI1)
  CALL CLOSE(FDI2)
  CALL CLOSE(FDO1)
#
```

```
9999 CONTINUE
9090 RETURN 1
     END
```

```
#--MSMTSD                    DRIVER
#
#IDENTIFICATION BEEX, RATFOR, VAX-11
#
#PURPOSE
#       DRIVER FOR COMMAND THAT GENERATES FREQUENCY LIMITED MSMTS
#       OVER THE MSMT DOMAIN
#
#ENTRY POINT MSMTSD(WORK,ERRET)
#
#*******************************************************************
#
 INCLUDE MACA1
 SUBROUTINE MSMTSD(WORK,*)
 IMPLICIT INTEGER (A - Z)
 INCLUDE GIPCOM
 INCLUDE ERROR
 INTEGER IDENT(.IDLENGTH)
 DIMENSION WORK(.ARB)
#
 EQUIVALENCE (NPPL,IDENT(.IDNPPL))
 EQUIVALENCE (NLIN,IDENT(.IDNLINS))
 EQUIVALENCE (MODE,IDENT(.IDMODE))
#
 CALL PPUSH('MSMTSD')
#                          OPEN INPUTFILE
 CALL RDKINL(FDI1,IDENT,.OLD,IEV,%9999)
 CALL CLOSE(FDI1)
#                          CHECK INPUTFILE
 IF(MODE^=.REALMODE) GOTO 9000
#
 NXT=1
 ILIN1=GETWP(NXT,.REALMODE,NPPL)
 ILIN2=GETWP(NXT,.REALMODE,NPPL)
#
 IF(.OK^=OSALOC(NXT)) GOTO 9010
#
 CALL COMTIN(%9999)
#
 CALL MSMTSA(FDI1,FDO1,FDO2,WORK(ILIN1),WORK(ILIN2),
             NLIN,NPPL,IEV,%9999)
#
 CALL PPOP
#
 RETURN
#                       ERROR CONDITIONS
 9000 CONTINUE
#                       ILLEGAL DATAMODE
 IEV=-2012
 GOTO 9999
#
 9010 CONTINUE
 IEV=OSGIEV(IEV)
 GOTO 9999
#
 9999 CONTINUE
 CALL CLOSE(FDI1)
 CALL CLOSE(FDO1)
 CALL CLOSE(FDO2)
#
 RETURN 1
 END
```

```
#  MSMTSA                          ALGORITHM
#
#IDENTIFICATION BEEX, RATFOR, VAX-11
#
#PURPOSE
#        ALGORITHM SECTION FOR COMMAND THAT GENERATES
#        FREQUENCY LIMITED MSMTS OVER THE MSMT DOMAIN
#
#ENTRY POINT MSMTSA(FDI,FDO1,FDO2,ILIN1,ILIN2,NLIN,NPPL,IEV,*)
#
#*********************************************************
#
 INCLUDE MACA1
 SUBROUTINE MSMTSA(FDI1,FDO1,FDO2,ILIN1,ILIN2,NLIN,NPPL,IEV,*)
 IMPLICIT INTEGER (A - Z)
 CHARACTER FDI1(.FDLENGTH), FDO1(.FDLENGTH), FDO2(.FDLENGTH)
 INTEGER IDENT(.IDLENGTH), JDENT(.IDLENGTH), LDENT(.IDLENGTH)
 REAL ILIN1(NPPL), ILIN2(NPPL), MSV, CONST
 INTEGER*4 DUMMY
#
 INCLUDE TTCOM
#
 CALL PPUSH('MSMTSA')
#                         OPEN INPUTFILE
 CALL CPYIDR(FDI1,IDENT,.OPNTMP,IEV,%9000)
 CALL DSCNAM('MSMTSA',IEV,%9000)
#
 DO I=6,19
 $(
 IF(I .EQ. 6 .OR. I .EQ. 7 .OR. I .EQ. 13 .OR. I .EQ. 14 .OR.
    I .EQ. 17 .OR. I .EQ. 19) $(
                                 X=IDENT(I)
                                 JDENT(I)=X
                                 LDENT(I)=X
                                $)
 $)
#                         OPEN OUTPUTFILE
 CALL COPYDS(FDO1,JDENT,IEV,%9000)
#
 YSL=1
 YSH=NLIN/2-1
 YSD=(YSL+YSH)/2
 CALL RNGETI('VERTICAL SIZE MSMT
DOMAIN?.',YSL,YSH,YSD,YS,IEV,%9000)  XSL=1
 XSH=NPPL/2-1
 XSD=(XSL+XSH)/2
 CALL RNGETI('HORIZONTAL SIZE MSMT
DOMAIN?.',XSL,XSH,XSD,XS,IEV,%9000)  NOIVL=0
 NOIVH=100000
 NOIVD=0
 CALL RNGETI('NOISE VARIANCE?.',NOIVL,NOIVH,NOIVD,NOIV,IEV,%9000)
#
 CONST=SQRT(12.*NOIV)
 IF(NOIV==NOIVD) CONST=0.0
 DUMMY=34589
 DO BLKN=1,NLIN
 $(
   CALL RREAD(FDI1,ILIN1,1,BLKN,IDENT,.WAIT,IEV,%9000)
   DO I=1,NPPL
   $(
     ILIN1(I)=ILIN1(I)+CONST*(RAN(DUMMY)-.5)
   $)
```

```
    CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
    CALL RREAD(FDI1,ILIN1,2,BLKN,IDENT,.WAIT,IEV,%9000)
    DO I=1,NPPL
    $(
      ILIN1(I)=ILIN1(I)+CONST*(RAN(DUMMY)-.5)
    $)
    CALL RWRITE(FDO1,ILIN1,2,BLKN,JDENT,.WAIT,IEV,%9000)
 $)
#
 BLKN=NLIN/2+1
 DO I=1,NPPL
 $(
   ILIN1(I)=0.
 $)
 ILIN1(1)=YS+.02
 ILIN1(2)=XS+.02
 CALL RWRITE(FDO1,ILIN1,1,BLKN,JDENT,.WAIT,IEV,%9000)
#
 CALL CLOSE(FDO1)
 CALL CLOSE(FDI1)
 CALL CPYIDR(FDI1,IDENT,.OPNTMP,IEV,%9000)
 CALL COPYDS(FDO2,LDENT,IEV,%9000)
 XSP1=XS+1
 NS=NPPL-XS+2
 NSM1=NS-1
 YSP1=YS+1
 MS=NLIN-YS+2
 MSM1=MS-1
#
 MSV=0.0
 DO BLKN=1,YS
 $(
   CALL RREAD(FDI1,ILIN1,1,BLKN,IDENT,.WAIT,IEV,%9000)
   CALL RREAD(FDI1,ILIN2,2,BLKN,IDENT,.WAIT,IEV,%9000)
   DO I=1,XS
   $(
     MSV=MSV+ILIN1(I)*ILIN1(I)+ILIN2(I)*ILIN2(I)
   $)
   DO I=NS,NPPL
   $(
     MSV=MSV+ILIN1(I)*ILIN1(I)+ILIN2(I)*ILIN2(I)
   $)
   DO I=XSP1,NSM1
   $(
     ILIN1(I)=0.0
     ILIN2(I)=0.0
   $)
   CALL RWRITE(FDO2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
   CALL RWRITE(FDO2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
 $)
 DO BLKN=MS,NLIN
 $(
   CALL RREAD(FDI1,ILIN1,1,BLKN,IDENT,.WAIT,IEV,%9000)
   CALL RREAD(FDI1,ILIN2,2,BLKN,IDENT,.WAIT,IEV,%9000)
   DO I=1,XS
   $(
     MSV=MSV+ILIN1(I)*ILIN1(I)+ILIN2(I)*ILIN2(I)
   $)
```

```
  DO I=NS,NPPL
  $(
    MSV=MSV+ILIN1(I)*ILIN1(I)+ILIN2(I)*ILIN2(I)
  $)
  DO I=XSP1,NSM1
  $(
    ILIN1(I)=0.0
    ILIN2(I)=0.0
  $)
  CALL RWRITE(FDO2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
  CALL RWRITE(FDO2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
$)
DO BLKN=YSP1,MSM1
$(
  DO I=1,NPPL
  $(
    ILIN1(I)=0.0
    ILIN2(I)=0.0
  $)
  CALL RWRITE(FDO2,ILIN1,1,BLKN,LDENT,.WAIT,IEV,%9000)
  CALL RWRITE(FDO2,ILIN2,2,BLKN,LDENT,.WAIT,IEV,%9000)
$)
MSV=MSV/((2*XS-1)*(2*YS-1))
WRITE(TTYOT,1010) MSV
1010 FORMAT(' MEAN SQUARE VALUE OF MSMTS= ',E10.4)
#
CALL PPOP
CALL CLOSE(FDI1)
CALL CLOSE(FDO1)
CALL CLOSE(FDO2)
RETURN
#                    ERROR CONDITIONS
9000 CONTINUE
CALL CLOSE(FDI1)
CALL CLOSE(FDO1)
CALL CLOSE(FDO2)
#
9999 CONTINUE
9090 RETURN 1
END
```

INFORMATION TRANSFERS PUBLISHED, PRESENTED, AND IN PREPARATION

"2-D Spectral Estimator Approaches to Enhanced Scene Resolution," presented by A.A. (Louis) Beex at the 1982 Virginia IEEE Conference and Exhibit (VACON '82), Hampton, VA. October 28 - 29, 1982.

"Iterative Reconstruction of Space-Limited Scenes from Noisy Frequency-Limited Measurements," presented and published by A.A. (Louis) Beex at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '83), Boston, MA. pp. 147 - 150, April 14 - 16, 1983.

"Soft Constraint Iterative Reconstruction from Noisy Projections," submitted for presentation and publication by A.A. (Louis) Beex at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '84), San Diego, CA. March 19 - 21, 1984.

"Reconstruction of Space-Limited Scenes from Noisy Frequency Domain Projections Using Soft Constraints," manuscript planned by A.A. (Louis) Beex for publication in the IEEE Transactions on Acoustics, Speech, and Signal Processing.

FIL